UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/037,666	01/03/2002	Sriram Vajapeyam	42390P11927	8277

8791 7590 05/30/2006

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 05/30/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/037,666	VAJAPEYAM ET AL.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 March 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-6,8-11,13-15 and 17-33* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-6,8-11,13-15 and 17-33* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *28 December 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *13 March 2006*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-6, 8-11, 13-15, and 17-33 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: RCE, IDS, and Amendment as received on 3/13/2006.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

4. Claims 1-6, 8-11, 13-15, 17-22, and 27-33 are rejected under 35 U.S.C. 102(e) as being anticipated by Park, U.S. Patent No. 6,988,190.

5. Referring to claim 1, Park has taught a logic circuit comprising:

a) data flow logic. The data flow logic is the logic which receives the fetched trace cache entries shown in Fig.6. Clearly, some logic exists which uses the information in the entries to perform some task.

b) control flow logic to select and fetch a trace descriptor for processing, the fetched trace descriptor including at least one dependency descriptor, the control flow logic to dispatch to the data flow logic a dependency descriptor including dependency information for an instruction

sequence and an address of the instruction sequence. See Fig.5-6. Trace descriptors are fetched and dispatched to data flow logic. The trace descriptors include an address of an instruction sequence (Fig.5, field 502) and a dependency descriptor including dependency information (Fig.5, fields 506 and 508). These fields constitute dependency information because the execution of the sequence is dependent on these fields. See column 4, lines 35-56.

c) the data flow logic coupled to the control flow logic to receive the dispatched dependency descriptor, to fetch the instruction sequence using the address from the received dependency descriptor, and to execute the instruction sequence according to the dependency information in the received dependency descriptor. As described above, the start address is used to fetch the sequence and the sequence is then executed based on the dependency information. See column 4, lines 35-56.

6. Referring to claim 2, Park has taught a logic circuit as described in claim 1. Park has further taught a storage area coupled to the control flow logic and the data flow logic, the storage area to store the dependency descriptor from the fetched trace descriptor by the control flow logic. See column 4, lines 35-44, and note that an address trace cache (storage area) holds the information of Fig.5 and Fig.6.

7. Referring to claim 3, Park has taught a logic circuit as described in claim 1. Park has further taught a storage area coupled to the control flow logic, the storage area to store trace descriptors. See column 4, lines 35-44, and note that an address trace cache (storage area) holds the information of Fig.5 and Fig.6.

8. Referring to claim 4, Park has taught a logic circuit as described in claim 1. Park has further taught a storage area coupled to the data flow logic, the storage area to store instructions

contiguously based on dependency information. The examiner asserts that instructions are inherently stored in some type of instruction memory. Furthermore, Fig.5 and Fig.6 show a start address and an end address for a routine, where the end address may be considered part of the dependency information. All instructions in that routine are stored contiguously in memory between those addresses.

9. Referring to claim 5, Park has taught a logic circuit as described in claim 1. Park has further taught a storage area coupled to the data flow logic and control flow logic, the storage area to store live-out data. Clearly, instructions write result data to some form of memory, whether it be to a stack, to main memory, or to a register file (which is the most common). The memory written to would hold live-out data, which is data used by subsequent instructions.

10. Referring to claim 6, Park has taught a logic circuit as described in claim 1. Park has further taught a storage area coupled to the control flow logic, the storage area to map dependency information. Fig.5 and Fig.6 show that dependency information is mapped to a certain instruction sequence by being store with information defining the locations of the instruction sequence.

11. Referring to claim 8, Park has taught a logic circuit as described in claim 1. Park has further taught that the trace descriptor includes aggregate live-in data for a plurality of dependency descriptors in the trace descriptor. See Fig.5, field 506, for instance. This data is updated as the count increases and is stored in the entry.

12. Referring to claim 9, Park has taught a logic circuit as described in claim 1. Park has further taught that the trace descriptor includes aggregate live-out data for a plurality of dependency descriptors in the trace descriptor. See Fig.5, field 506, for instance. This data is

updated as the count increases and is used by circuitry external to the address trace cache to determine whether the sequence of instructions is done executing.

13. Referring to claim 10, Park has taught a computer system comprising:

a) at least one memory device to store trace descriptors and instruction sequences. See Fig.5 and note the format of entries in a address trace cache. In addition, it is inherent that instructions are stored in some form of instruction memory.

b) a bus coupled to the at least one memory device. A bus must inherently exist if data/instructions are to be retrieved from the memory device(s).

c) control flow logic to select and fetch one of the trace descriptors, the fetched trace descriptor including a plurality of dependency descriptors having locations of corresponding instruction sequences and having dependency information for corresponding instruction sequences. See Fig.5 and Fig.6 and note that any combination of fields in the entry may be considered a dependency descriptor. Also, the entry includes a start address from which an instruction sequence is fetched. Fields 506 and 508 at least constitute dependency information because the execution of the sequence is dependent on these fields. See column 4, lines 35-56.

d) data flow logic coupled to the control flow logic to receive a dependency descriptor dispatched from the control flow logic, to fetch an instruction sequence corresponding to the received dependency descriptor, and to execute the fetched instruction sequence according to dependency information in the received dependency descriptor. See column 4, lines 35-56. Instructions are fetched based on the start address and executed based on the dependency descriptor (i.e., they are executed as many times as is required by the descriptor).

14. Referring to claim 11, Park has taught a computer system as described in claim 10. Park has further taught an issue window coupled between the control flow logic and the data flow logic, the issue window to store the dependency descriptor dispatched from the control flow logic. The information in the trace descriptor, including the dependency descriptor(s) must be sent somewhere to be analyzed. Since the information causes instructions to be issued, it can be said that an "issue window" holds this information.

15. Referring to claim 13, Park has taught a computer system as described in claim 10. Park has further taught that the at least one memory device is to store an instruction sequence contiguously based on dependency information. The examiner asserts that instructions are inherently stored in some type of instruction memory. Furthermore, Fig.5 and Fig.6 show a start address and an end address for a routine, where the end address may be considered part of the dependency information. All instructions in that routine are stored contiguously in memory between those addresses.

16. Referring to claim 14, Park has taught a computer system as described in claim 10. Park has further taught a storage area coupled to the data flow logic and control flow logic, the storage area to store live-out data. Clearly, instructions write result data to some form of memory, whether it be to a stack, to main memory, or to a register file (which is the most common). The memory written to would hold live-out data, which is data used by subsequent instructions.

17. Referring to claim 15, Park has taught a computer system as described in claim 10. Park has further taught a storage area coupled to the control flow logic, the storage area to map dependency information. Fig.5 and Fig.6 show that dependency information is mapped to a

certain instruction sequence by being store with information defining the locations of the instruction sequence.

18. Referring to claim 17, Park has taught a computer system as described in claim 10. Park has further taught that the fetched trace descriptor includes aggregate live-in information for dependency descriptors in the fetched trace descriptor. See Fig.5, field 506, for instance. This data is updated as the count increases and is stored in the entry.

19. Referring to claim 18, Park has taught a computer system as described in claim 10. Park has further taught that the fetched trace descriptor includes aggregate live-out information for dependency descriptors in the fetched trace descriptor. See Fig.5, field 506, for instance. This data is updated as the count increases and is used by circuitry external to the address trace cache to determine whether the sequence of instructions is done executing.

20. Referring to claim 19, Park has taught a computer system as described in claim 10. Park has further taught that dependency information in the received dependency descriptor includes live-in and live-out data. See Fig.5 and Fig.6, and note that the data held in field 506 is live-in data because it only lives within the entry itself. The data held in field 508 is live-out data because the count is pulled and modified by circuitry outside of the address cache.

21. Referring to claim 20, Park has taught a method of processing instructions comprising:

a) selecting and fetching a trace descriptor in accordance with program control flow. See Fig.5 and Fig.6.

b) identifying from the fetched trace descriptor a dependency descriptor including dependency information for a set of instructions and an address of the set of instructions. See Fig.5 and

Fig.6. Note that a start address (field 502) of a sequence of instructions as well as dependency information (fields 506 and 508) exists.

c) dispatching the dependency descriptor for execution. After fetching the descriptor, it must be dispatched somewhere for extraction and analysis. This is done for execution of the associated sequence of instructions.

d) fetching the set of instructions using the address from the dispatched dependency descriptor. Again, field 502 of the descriptor is used to fetch the sequence.

e) executing the set of instructions according to dependency information in the dispatched dependency descriptor. See column 4, lines 35-56.

22. Referring to claim 21, Park has taught a method as described in claim 20. Park has further taught updating live-out data in a storage area. Clearly, instructions write result data to some form of memory, whether it be to a stack, to main memory, or to a register file (which is the most common). The memory written to would hold live-out data, which is data used by subsequent instructions.

23. Referring to claim 22, Park has taught a method as described in claim 20. Park has further taught:

a) storing the identified dependency descriptor from a control flow logic into a storage area. Since the dependency descriptor is updated by "control flow" logic (column 4, lines 35-56), then the control flow logic will store the updates in the address cache.

b) reading the dependency descriptor out of the storage area into the data flow logic. The entries of Fig.5 and Fig.6 are read and used in the execution process.

24. Referring to claim 27, Park has taught a method as described in claim 20. Park has further taught that the selecting comprises predicting a next trace descriptor to process. See the summary of invention section of Park.

25. Referring to claim 28, Park has taught a method of processing instructions comprising:

a) selecting and fetching a trace descriptor in accordance with program control flow. See Fig.5 and Fig.6.

b) identifying from the fetched trace descriptor a dependency descriptor including dependency information for a set of instructions and an address of the set of instructions. See Fig.5 and Fig.6. Note that a start address (field 502) of a sequence of instructions as well as dependency information (fields 506 and 508) exists.

c) dispatching the dependency descriptor for execution. After fetching the descriptor, it must be dispatched somewhere for extraction and analysis. This is done for execution of the associated sequence of instructions.

d) fetching the set of instructions using the address from the dispatched dependency descriptor. Again, field 502 of the descriptor is used to fetch the sequence.

e) executing the set of instructions according to dependency information in the dispatched dependency descriptor. See column 4, lines 35-56.

26. Referring to claim 29, Park has taught a medium as described in claim 28. Park has further taught that the operations further comprise updating live-out data in a storage area. Clearly, instructions write result data to some form of memory, whether it be to a stack, to main memory, or to a register file (which is the most common). The memory written to would hold live-out data, which is data used by subsequent instructions.

27. Referring to claim 30, Park has taught a medium as described in claim 28. Park has further taught:

a) storing the dependency descriptor in an issue window by control flow logic. The information in the trace descriptor, including the dependency descriptor(s) must be sent somewhere to be analyzed. Since the information causes instructions to be issued, it can be said that an "issue window" holds this information.

b) reading the dependency descriptor out of the issue window into data flow logic. The information must be read to be analyzed.

28. Referring to claim 31, Park has taught a logic circuit as described in claim 1. Park has further taught that the fetched trace descriptor includes a plurality of dependency descriptors having addresses of corresponding instruction sequences and having dependency information for corresponding instruction sequences. See Fig.6.

29. Referring to claim 32, Park has taught a logic circuit as described in claim 1. Park has further taught that the dependency information includes live-in information. See Fig.5, field 506, for instance. This data is updated as the count increases and is stored in the entry, where it "lives".

30. Referring to claim 33, Park has taught a logic circuit as described in claim 1. Park has further taught that the dependency information includes live-out information. See Fig.5, field 506, for instance. This data is updated as the count increases and is used by circuitry external to the address trace cache to determine whether the sequence of instructions is done executing.

Claim Rejections - 35 USC § 103

31. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

32. Claims 23-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Park in view of Arimilli et al., U.S. Patent No. 6,427,204 (as applied in the previous Office Action and herein referred to as Arimilli).

33. Referring to claim 23, Park has taught a method as described in claim 20. Park has not taught that the fetching of a set of instructions is completed just in time for execution. However, Arimilli has taught such a concept. See column 3, lines 1-17. Note that Arimilli has taught that this is a more efficient way of fetching because instructions are only delivered when they are actually needed and pipeline bubbles are prevented. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Park such that instructions are fetched just-in-time, as taught by Arimilli.

34. Referring to claim 24, Park has taught a method as described in claim 20. Although Park has not taught that the instructions are out of order, Arimilli has taught such a concept. See column 1, line 61, to column 2, line 6. Note that the use of resources and efficiency are maximized with out-of-order execution. In addition, out-of-order execution allows for a reduction in stalling. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Park to include instructions that are out-of-order, as taught by Arimilli.

35. Claims 25-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Park in view of Witt et al., U.S. Patent No. 6,018,798 (as applied in the previous Office Action and herein referred to as Witt).

36. Referring to claim 25, Park has taught a method as described in claim 21. Park has not explicitly taught updating the architectural state using the data in the storage area. However, Witt has taught the concept of having a speculative register file (future file 88, Fig.3) and an actual register file (Fig.3, component 102). The speculative register file holds the most current state of the machine (values determined via speculative execution) and by doing this, instructions may be executed speculatively. Once it is determined that instructions are no longer speculative, the speculative results are made architectural results by writing them to the actual register file. See column 12, line 66, to column 13, line 45. This is a known concept in the art. In essence, this scheme allows for speculative execution which is a method of executing instructions before it is known that they should execute (they are predicted to execute). This maximizes efficiency if they indeed were to execute (predicted correctly). As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Park such that the architectural state is updated using the data in the speculative storage.

37. Referring to claim 26, Park in view of Witt has taught a method as described in claim 25. Witt has further taught recovering an earlier architectural state after a misprediction using data in the storage area. See column 18, lines 54-67, and note that after a misprediction, a previous state is achieved by copying actual values into the future file (so that the speculative values are

correct). Consequently, by using this newly written data, the system recovers an earlier architectural state.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
May 17, 2006

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100